A Trajectory Planner for Autonomous Structural Assembly

Catharine L. R. McGhan[*]
University of Maryland, College Park, Maryland, 20742

Autonomous robotic space construction requires robust scheduling routines to decompose and order assembly activities and an efficient trajectory planner to study the activities and determine how they can be physically accomplished. Task schedulers search through possible combinations of abstract "assemble-part-x" tasks, relying upon the path planner to compute a valid and efficient trajectory that completes the task and to provide an estimate of the cost of moving along that trajectory in terms of fuel or power requirements. This paper presents a path planner for robotic assembly of a truss structure composed of interchangeable beams and hubs to which beams can be attached. Because initial testing will be performed in a neutral buoyancy environment, the dynamics are based on underwater motion. A free-flying robot and the structural elements it carries to assembly sites are modeled as simple shapes (e.g., spheres, cylinders) with known inertia and drag properties. Each path is optimized over a cost function that currently includes time and fuel (power) use, which in turn provides a scheduler with a comparative cost estimate useful for choosing between multiple task configurations and combinations. Physical parameters are derived from the University of Maryland's Supplemental Camera and Maneuvering Platform Space Simulation Vehicle (SCAMP SSV) robot and six-element EASE truss structure, the latter previously assembled by astronauts in both space and neutral buoyancy underwater environments. Tests were conducted over a variety of initial and final structural element configurations, demonstrating that the proposed computationally-efficient path planning strategy provides practical and reasonable trajectories and costs.

Nomenclature

a	=	translational acceleration vector, $[a_x\ a_y\ a_z]$
C_{dv}	=	coefficient of drag, translational motion
$C_{d\omega}$	=	coefficient of drag, rotational motion
d	=	translational distance vector, $[d_x\ d_y\ d_z]$
F_{MAX}	=	maximum force output of a vehicle thruster
I	=	moment of inertia
J	=	cost of efficiently completing a task
L_B	=	beam length
m_B	=	beam mass
m_S	=	SCAMP SSV mass
${}^{x}p_{nE}$	=	vector p of description n of object E in coordinate frame x, (I = inertial, S = SSV, B = beam, H = hub)
R_B	=	beam radius
R_S	=	SCAMP SSV radius
t	=	time
$T_{x^\wedge y}$	=	transformation matrix from x to y, $[T_0\ T_1\ T_2]$
u	=	fuel vector for each thruster, $[u_1\ u_2\ u_3\ u_4\ u_5\ u_6]$
v	=	translational velocity, vector $[v_x\ v_y\ v_z]$
W	=	weighting factor
α	=	rotational acceleration vector, $[a_x\ a_y\ a_z]$
θ	=	rotational angle vector, $[\theta_x\ \theta_y\ \theta_z]$
ω	=	rotational velocity vector, $[\omega_x\ \omega_y\ \omega_z]$

[*] Undergraduate Researcher, Department of Aerospace Engineering, Space Systems Lab, Bldg. #382, College Park, Maryland, 20742, AIAA student member.

I. Introduction

Autonomous space construction is required for large-scale projects, and significant progress must be made in the field of robotics before it can become a cost-effective reality. Advances in simulation and planning should take place in conjunction with technological advances and growing commercial interests. A simple first approximation of object and vehicle movement can be attained in computer simulations, but real-world testing is also important, and neutral buoyancy testing is a relatively easy way of experimenting with the same three-dimensional, six degree-of-freedom (DOF) motion available in a space environment. Most discrete path planners and real-time task schedulers do not take into account the amount of time or fuel that the motion of a vehicle 'spends' carrying out its tasks, and are optimized instead to identify the shortest obstacle-free path. The solutions found by these algorithms are therefore not truly optimized in a real-world sense, since fuel is in short supply on any spacecraft (once it is used it is gone; currently there are no viable spacecraft refueling technologies) and is a driving factor in the lifetime of the vehicle. The time to completion of a task can also be critical in collision avoidance and station upkeep. The inefficient completion of a large number of tasks would waste valuable time and fuel resources. The cost function weights used also have a significant impact on the outcome of the optimal solution.

This paper presents a simple model for robotic structural assembly and uses a rule-based trajectory planner to determine the execution of these tasks underwater. This work provides a foundation for the development of more complex path planners that guarantee obstacle avoidance while optimizing over time and fuel varying relative weighting factors. First, a description of the problem and the assumptions made are provided, along with a review of the equations of motion and dynamics governing the vehicle and objects modeled in this paper. Then, the trajectory generation algorithm is presented, followed by a case study, final conclusions, and a discussion of future work.

II. Problem Description

A. Overview

Consider a task scheduler that uses Hierarchical Task Network planning and ordered task decomposition, such as the SHOP-2 planner.[1] Such an algorithm can be used to solve generalized scenarios with multiple interchangeable parts and order assembly tasks. Given interchangeable parts A and B, part A can be connected to the first section and part B to the second section, or vice-versa, so several combinations may be found as workable solutions. However, this does open up the problem of deciding which schedule to use over the others. To find an optimal solution, an analysis of the costs of each assembly task must be made. This typically requires a cost function with weights associated with the "priority" of particular physical factors in conjunction with a dynamic trajectory planner that keeps track of the movement of the vehicle and the costs incurred (in terms of fuel, time, etc.), to give a numerical value J that can be used to determine the relative cost of each assembly task. Generally,

$$\begin{aligned} J &= \sum W_i \cdot X_i \\ &= W_1 \cdot \int dt + W_2 \cdot \int \|u\| \cdot dt \end{aligned} \tag{1}$$

where X_i are normalized cost terms. The algorithms used to determine and keep track of the dynamic state should be simple, however, to keep the runtime of the path planner short: lengthening the runtime of the path planner would exponentially increase the runtime of the task scheduler that calls it. Figure 1 shows the modules required for a complete task scheduling / trajectory planning system, illustrating how the trajectory planner fits into the overall system.

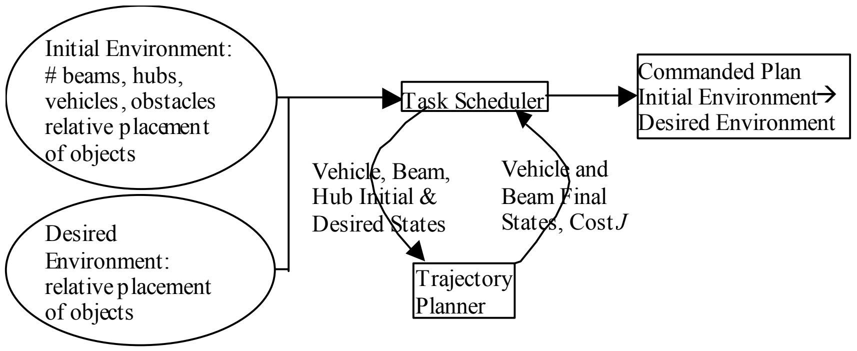
Initial Environment: # beams, hubs, vehicles, obstacles relative placement of objects
Desired Environment: relative placement of objects
Task Scheduler
Commanded Plan Initial Environment→ Desired Environment
Vehicle, Beam, Hub Initial & Desired States
Vehicle and Beam Final States, Cost J
Trajectory Planner

Figure 1: Autonomous Structural Assembly Planning Architecture.

B. Problem Statement

For a given task scheduler that requires a general measure J of the cost of efficiently completing a task, the goal is to implement a trajectory planner that constructs feasible and efficient continuous-time trajectories and assesses their cost. For this work, the EASE[2] (Experimental Assembly of Structures in EVA) structure is assembled using the SCAMP SSV[3] (Supplemental Camera And Maneuvering Platform Space Simulation Vehicle) free-flying neutral buoyancy robot. The trajectory planner computes an efficient route based on time and fuel consumption. Each construction task involves two sub-tasks: an "SSV travel to beam" event, and an "SSV fly beam to hub" event. Figure 2 shows the vehicle and objects, and Fig. 3 describes the general movement of the objects during the construction task.

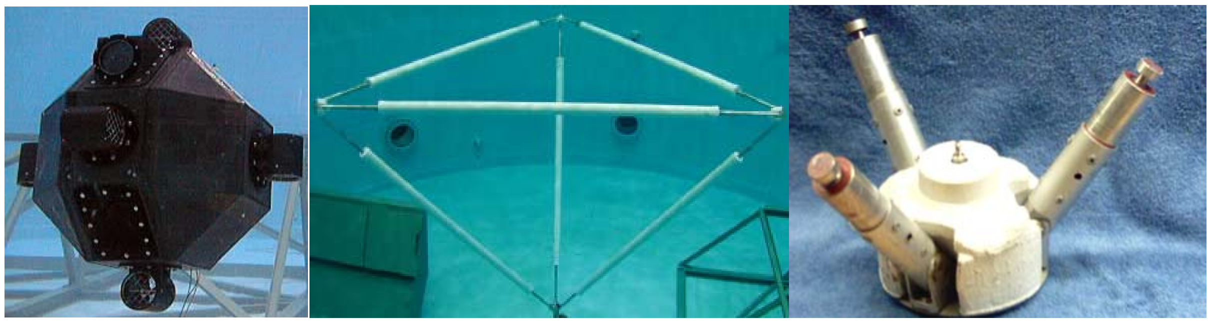
Figure 2: Images of SCAMP SSV, The Fully-Constructed EASE Structure, and a Hub

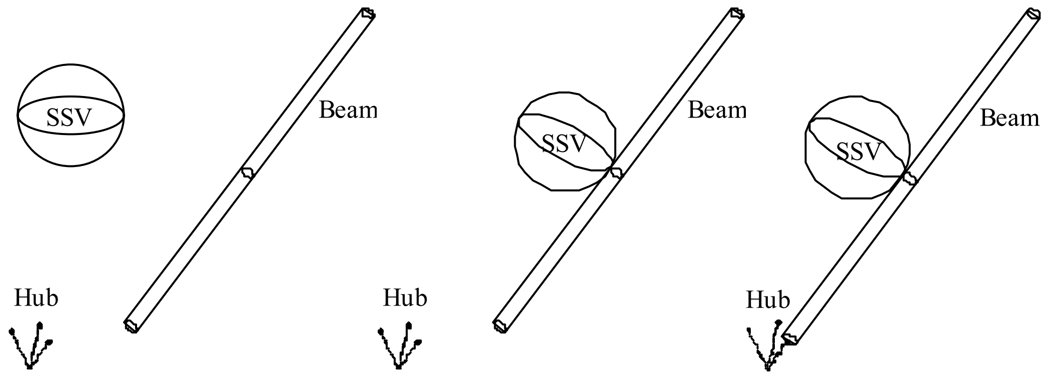

Figure 3: Initial State, SSV→Beam, SSV-Beam→Hub

A task space consists of an inertial frame of reference that encompasses one vehicle, one beam, and one hub, with no other obstacles impinging upon this space. In order to simplify the equations of motion, a constant F_{MAX}, maximum thruster output, is applied for translational motion, and a corresponding maximum moment limit is applied for rotational motion. Translational and rotational motion is decoupled to further simplify the equations of motion (the more complex Magnus-Robins[4] effect – the lift created by the rotation of a cylinder or sphere – is not used to the planner's advantage). Fuel/power consumption is measured by the integration of all thruster forces over time. Total time elapsed, t_{total}, is:

$$t_{total} = t_S + t_{SB} \tag{2}$$

where t_S is the time it takes SSV to move to the beam, and t_{SB} is the time it takes SSV to move the beam to the hub.

1. Assumptions

- Object states are precisely known and do not change during the computation).
- Vibrations are negligible.
- The objects and the vehicle are initially at rest.
- The beam, hub, and SCAMP SSV vehicle have uniform mass distributions and are neutrally buoyant in translation and rotation.
- Laminar flow is assumed (which leads to linear drag equations), and all wake flow is negligible.

- SCAMP SSV can be modeled as a perfect sphere; beams can be modeled as cylindrical tubes.
- The beam experiences drag along its longitudinal axis, but not over or along the circular cross-section ends.
- The identical propeller-driven thrusters on SCAMP SSV instantaneously deliver commanded thrust loads.
- SCAMP SSV has a single magical contact point that when aligned properly attaches the vehicle rigidly to a beam; similarly, a beam can be attached to a hub once aligned.
- Docking (SSV to beam or beam to hub) occurs without force application to the target.

III. Trajectory Planner Implementation

A. General Implementation Procedure

A trajectory planner has been implemented for determining EASE beam transport paths with SCAMP SSV. The procedure to assemble each structural element consists of two main steps: SSV travels to the beam, and the SSV-beam pair travels to the hub. The trajectory planner (and built-in cost function) is input data detailing the initial state (translation and rotation) of SSV, the beam (with one end specified as the candidate for mating), and the hub (with one beam contact point specified as the mating port). It is assumed that the objects and the vehicle are initially at rest. A discrete set of valid SSV-beam mating points is identified (see Fig. 4), and then the path planner searches over each of these contact points over the full transport path to find the minimum-cost solution. During a full transport path, the vehicle flies to the beam and mates to it, then the vehicle-beam pair moves to the hub contact point. Time and fuel consumption are tallied over all trajectory segments, and the SSV-beam contact point and thrust scenario with minimum total cost is selected as the solution. The total minimum cost *J*, along with the final state data, is then output to the task scheduler.

B. Data Input

The state data given to the cost function includes the coordinate system at the center of mass of each object – vehicle, beam, hub – and contact point(s) on each object that correspond to the location of the physical connection between the two mated objects. Coordinate systems are defined by a vector that gives the translational offset of the origin of the system and a rotation matrix that describes the orientation of the system relative to an inertial frame. ${}^{S}p_{contactS}$ on SCAMP SSV can rigidly attach to any point on the beam and is defined as a point a distance R_S from the center of the vehicle along the local vehicle x-axis (see Fig. 5). ${}^{B}p_{contactB}$ on the beam corresponds to one of the two endpoints that attaches to the hub (see Fig. 4), while one of the three contact points on the hub ${}^{H}p_{contactH}$ is that point (see Fig. 8). The rotation of the beam about its longitudinal axis does not matter in terms of attaching the beam to the hub, so the x'-y' axes of the contact point system are not statically defined – only the x'-y' plane is defined. A similar argument is true for the vehicle about its contact point when attaching to the beam.

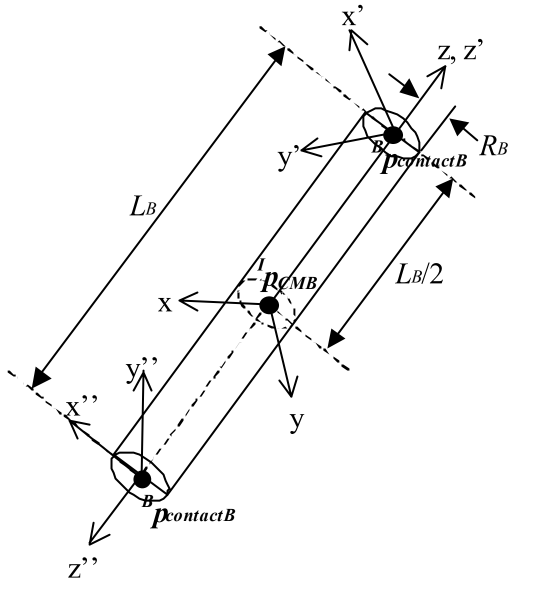

Figure 4: Beam coordinate systems and points.

C. Selection of Beam Mating Points

The mating point for the vehicle to attach to the beam can be found analytically for the first half of the SSV-to-beam trajectory. However, because the drag characteristics can change significantly depending on where the vehicle attaches itself to the beam, and because the orientation of the vehicle determines the type of movement and thrust required to move the pair, there is no simple analytical method for determining the best place for the vehicle to attach itself to the beam for the entire assembly process. The beam is divided into sections, with one 'ring' of points per section. In the initial testing, three rings were searched – one at each end around ${}^{B}p_{contactB}$ and one in the middle around ${}^{B}p_{CMB}$, the center of mass point of the beam – to obtain three points ${}^{S}p_{attachS}$ at which ${}^{S}p_{contactS}$ may attach. An additional point could be added to the end without the contact point, since the object being moved is known to be a single beam unattached to any other structure, but because the current planner has no object avoidance code this is not implemented. In order to test both cases (attaching each of the open ends to the hub), the cost function is run twice, once with the first contact point, and again with the second. To find the inertial representation of each ${}^{S}p_{attachS}$, the local points ${}^{B}p_{contactB}$ are transformed into inertial space:

$$ {}^{I}p_{contact,i} = T_{B^\wedge I} \cdot [0 \quad 0 \quad L_B \cdot i/2] + {}^{I}p_{CMB}, i = -1 \rightarrow 1 \tag{3}$$

Technically, we don't want to find the closest points on the beam to SSV to which ${}^{S}p_{contactS}$ attaches – we want to find the translational motion, the closest points in SSV space ${}^{S}N_S$ where ${}^{I}p_{CMS}$ will be located after translating, on a ring which is R_S+R_B outward from ${}^{I}p_{contactB,i}$. To find the corresponding ${}^{S}N_S$ points for each ${}^{I}p_{contactB,i}$:

$$\begin{aligned}
{}^{B}N_{1,i} &= T_{I^\wedge B} \cdot (({}^{I}p_{CMS} - {}^{I}p_{contactB,i}) - {}^{I}p_{CMB}) \\
{}^{B}N_{2,i} &= [{}^{B}N_{1x,i} \quad {}^{B}N_{1y,i} \quad 0] \\
{}^{B}N_{3,i} &= {}^{B}\hat{N}_{2,i} \cdot (R_B + R_S) \\
{}^{B}p_{contactB,i} &= T_{I^\wedge B} \cdot ({}^{I}p_{contactB,i} - {}^{I}p_{CMB}) \\
{}^{B}N_{4,i} &= [{}^{B}N_{3x} \quad {}^{B}N_{3y} \quad {}^{B}p_{contactBz,i}] \\
{}^{S}N_{S,i} &= T_{I^\wedge S} \cdot (((T_{B^\wedge I} \cdot {}^{B}N_{4,i}) + {}^{I}p_{CMB}) - {}^{I}p_{CMS})
\end{aligned} \tag{4}$$

Once the ${}^{S}N_{S,i}$ points have been found, the rest of the assembly process is repeated for each attach point case.

D. SCAMP SSV Movement

The vehicle trajectory planner solves decoupled equations of motion by first translating then rotating the vehicle. Computation of the optimal coupled 6-DOF motion is left for future work. Implementing translation ${}^{S}N_S$ before rotation does not have any effect on drag, since SSV is being modeled as a perfect sphere and will have the same drag regardless of the direction in which it moves. Translation alters the SSV center-of-mass position ${}^{I}p_{CMS}$ by ${}^{S}N_S$. The vehicle then rotates so that the ${}^{S}p_{contactS}$ point is perpendicular to the surface of the beam.

To find the axis about which SSV rotates, and the angle that it rotates through, we follow Eqs. (5):

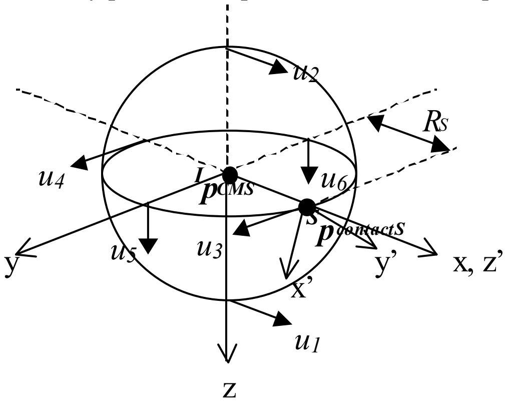

Figure 5: SCAMP SSV thruster force directions, coordinate systems, and points.

$$\begin{aligned}
{}^{S}p_{newcontactS} &= {}^{S}N_S + {}^{S}p_{contactS} \\
{}^{S}p_{contactlineS} &= T_{I^\wedge S} \cdot ({}^{I}p_{contactB} - {}^{I}p_{CMS}) \\
{}^{S}axis &= {}^{S}\hat{p}_{newcontactS} \times {}^{S}\hat{p}_{contactlineS} \\
{}^{S}angle &= \cos^{-1}({}^{S}\hat{p}_{newcontactS} \bullet {}^{S}\hat{p}_{contactlineS})
\end{aligned} \tag{5}$$

For SCAMP SSV alone, the vehicle dynamics are rather simple, and because we assume that the thrusters will nominally operate at maximum thrust, the velocity of the vehicle can be characterized by either: an approximately constant-slope ramp up to terminal velocity, a drift or maximum deceleration ramp to stop, and a level terminal velocity joining segment. While the vehicle ramps up to and maintains

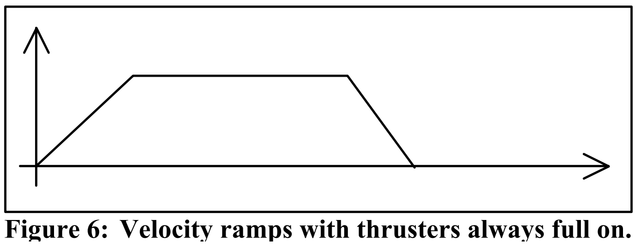
Figure 6: Velocity ramps with thrusters always full on.

terminal velocity, it is assumed that the thrusters are supplying maximum thrust. However, when the vehicle slows down it can do so in two ways: it can either turn on the thrusters full reverse or it can let drag do all the work to slow it to a halt (see Figs. 6, 7).

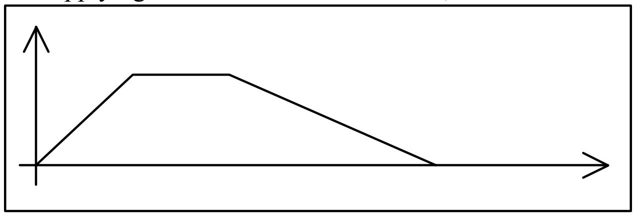
Figure 7: Velocity ramps with thrusters full on until the end.

Once these velocity ramps are determined, the solution is straightforward. For the rotational motion case, all one has to do is look at the angles the vehicle moves through during each ramp – if the total angle necessary is equal to or greater than the addition of the ramp up and each ramp down, then the extra angle segment necessary can be acquired by 'constant holding' the vehicle at terminal velocity as long as necessary. If the total angle necessary is less than the addition of the two ramps, then a ramp-fitting exercise occurs, until a case is found where both the ramp up finish and ramp down start have matching speeds and the total angle of the combined ramp movements equals the total angle necessary. The method used to find these ramps comes from forward iterations of the SSV dynamic equations shown in Eqs. (6). The equations of motion are derived from kinematics and an SSV document.[3]

$$\begin{aligned}
&\sum \tau = I_S \cdot \alpha = F \cdot R - C_{d\omega S} * \omega^2 \\
&\rightarrow \alpha_i = (F_{MAX} \cdot 2 \cdot R_S - C_{d\omega S} * \omega_{i-1}^2) / I_S \\
&\omega_i = \omega_{i-1} + \alpha_{i-1} \cdot \Delta t \\
&\theta_i = \theta_{i-1} + \omega_{i-1} \cdot \Delta t + 0.5 \cdot \alpha_{i-1} \cdot \Delta t^2
\end{aligned} \tag{6}$$

For the acceleration ramp, the initial velocities are zero. For the maximum-thrust decelerating ramp, the initial condition for ω is the final condition from the acceleration ramp, $F_{MAX}*2*R_S$ is negative, and the iteration continues until $\omega_i = 0$. For the drag-only deceleration ramp, the initial condition for ω is the final condition from the full-on ramp up movement (the others are zeroed), $F_{MAX}*2*R_S$ is removed, and the while loop is run until $\omega_i = 0$.

A similar exercise occurs for the translational motion; all that is required are a few simple substitutions, and one solves for the translational motions of the vector instead. Eqs. (7) describe the acceleration segment.

$$\begin{aligned}
&\sum F = m_S \cdot a = F - C_{dvS} * v \\
&\rightarrow a_i = (F_{MAX} \cdot 2 - C_{dvS} * v_{i-1}^2) / m_S \\
&v_i = v_{i-1} + a_{i-1} \cdot \Delta t \\
&d_i = d_{i-1} + v_{i-1} \cdot \Delta t + 0.5 \cdot a_{i-1} \cdot \Delta t^2
\end{aligned} \tag{7}$$

Once this is accomplished, the times are known, the forces exerted during translation are directly known, and the moments exerted during rotation are known and the forces fall out of those easily as well. From the forces and the amount of time the thrusters are on, the fuel consumption can be found for each force u_i in Newtons exerted for a known period of time t_i in seconds from Eq. (1).

E. Determination of Destination Points on Hub

The hub is modeled as an inertial-frame-fixed object, and therefore the specifics of its mass, inertia, etc. need not be defined. Figure 8 shows just such a general structure. There are three possible mating points on each hub, the choice of which is specified by the task scheduler as the final beam/hub state. The coordinate system of the contact point has the *z'*-axis running from the center of mass through the contact point, and the *x'*-*y'* plane is perpendicular to that; the *x'*-*y'* plane does not need to be strictly defined, because the beam *x'*-*y'* plane of its contact point also does not need to be strictly defined. The ${}^{H}p_{contactH}$ points are all fixed relative to ${}^{I}p_{CMH}$: all *z'*-axes are

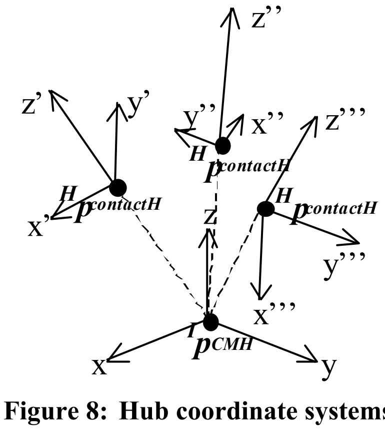

Figure 8: Hub coordinate systems and points.

bent down 30° from the z-axis; the projection of the first ${}^{H}p_{contactH}$ point falls on the x-axis, and all ${}^{H}p_{contactH}$ points are 120° offset from each other in the x-y plane; and the distance from each ${}^{H}p_{contactH}$ to ${}^{I}p_{CMH}$ is 0.381 m (15 in).

F. Combined SCAMP SSV-Beam Movement

Because thrust is not applied symmetrically about the center of mass of the combined SSV-beam object, the inertia matrix is not straightforward and the forces are not evenly distributed. This makes it difficult to decouple translational and rotational motion, so some broad restrictions are made.

The SSV-beam path planner solves the equations of motion by first rotating then translating the pair, and it finds the angle and axis in a similar way as before. It should be noted that, whereas before the ramps were the same for every run, these ramps change depending on each new placement of the vehicle along the beam, and must be resolved again every time. It should also be noted that this may put the beam in a high drag configuration and that the coefficients of translational and rotational drag for the cylinder will change depending on which ${}^{I}p_{contactB,i}$ is being solved.

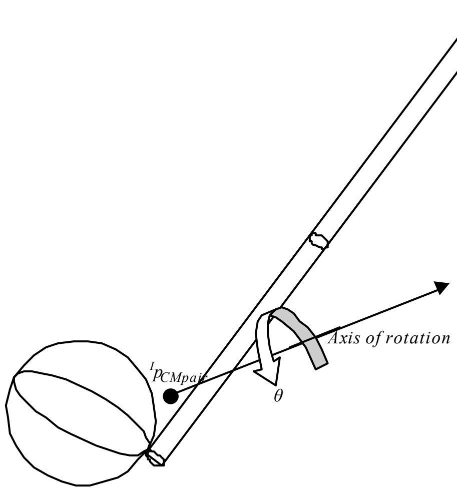

Figure 9: Axis and angle of rotation

First, the equations need to be simplified as much as possible. If the motion is taken about the principal axes of inertia (the origin is at the center of mass[5], and the orientation of the system is determined by solving an eigenvalue problem[6]), then the inertia matrix becomes diagonalized again. Setting the applied moments equal to an upper limit leaves three fewer variables to solve (though it does make the solution less optimized). Using the first equation from Eqs. (6) and a dynamics text[6] gives:

$$\begin{aligned}
\sum M_x &= I_x \cdot \alpha_x - (I_y - I_z)\cdot \omega_y \cdot \omega_z = M_{MAX} - (C_{d\omega S} + C_{d\omega B}) * \omega_x^2 \\
\sum M_y &= I_y \cdot \alpha_y - (I_z - I_x)\cdot \omega_z \cdot \omega_x = M_{MAX} - (C_{d\omega S} + C_{d\omega B}) * \omega_y^2 \\
\sum M_z &= I_z \cdot \alpha_z - (I_x - I_y)\cdot \omega_x \cdot \omega_y = M_{MAX} - (C_{d\omega S} + C_{d\omega B}) * \omega_z^2
\end{aligned} \tag{9}$$

To simplify further, the rotational motion can be solved by finding the moment of inertia about an arbitrary axis, I_{CMaxis}, as before, rather than in x-y-z components:[6]

$$I_{CMaxis} = I_{xx} \cdot u_x^2 + I_{yy} \cdot u_y^2 + I_{zz} \cdot u_z^2 - 2\cdot I_{xy} \cdot u_x \cdot u_y - 2\cdot I_{yz} \cdot u_y \cdot u_z - 2\cdot I_{zx} \cdot u_z \cdot u_x \tag{10}$$

with $u = [u_x\ u_y\ u_z]$ as the unit vector along the axis of rotation (see Fig. 9); this leaves one equation with one unknown to be solved iteratively, after the constant value M_{MAX} is applied.

Combining Eqs. (9) and (10) gives:

$$\begin{aligned}
&\sum M = I_{CMaxis} \cdot \alpha = M_{MAX} - (C_{d\omega S} + C_{d\omega B}) * \omega^2 \\
&\rightarrow \alpha_i = (M_{MAX} - (C_{d\omega S} + C_{d\omega B}) * \omega_{i-1}^2) / I_{CMaxis} \\
&\omega_i = \omega_{i-1} + \alpha_{i-1} \cdot \Delta t \\
&\theta_i = \theta_{i-1} + \omega_{i-1} \cdot \Delta t + 0.5 \cdot \alpha_{i-1} \cdot \Delta t^2
\end{aligned} \tag{11}$$

Once the ramps are computed and M_{MAX} has been converted to the SSV coordinate frame ($M = [M_x\ M_y\ M_z]$ is given), the moments must be solved to find the forces each thruster exerts for pure rotation only (the simplifications are due to the thruster force vectors, see Fig. 5):

$$\begin{aligned}
\sum M &= r_1 \times u_1 + r_2 \times u_2 + r_3 \times u_3 + r_4 \times u_4 + r_5 \times u_5 + r_6 \times u_6 \\
M_x &= 0 + 0 - r_{z3} \cdot u_3 + r_{z4} \cdot u_4 + r_{y5} \cdot u_5 - r_{y6} \cdot u_6 \\
M_y &= r_{z1} \cdot u_1 - r_{z2} \cdot u_2 + 0 + 0 - r_{x5} \cdot u_5 - r_{x6} \cdot u_6 \\
M_z &= -r_{y1} \cdot u_1 + r_{y2} \cdot u_2 + r_{x3} \cdot u_3 - r_{x4} \cdot u_4 + 0 + 0
\end{aligned} \tag{12}$$

where $r_i = [r_{xi}\ r_{yi}\ r_{zi}]$ is the vector from the center of mass of the object combination to thruster u_i. In order to have pure rotation only, the forces must balance:

$$\begin{aligned}
u_1 + u_2 &= 0 \\
u_3 + u_4 &= 0 \\
u_5 + u_6 &= 0
\end{aligned} \tag{13}$$

then the equations can be solved symbolically and entered directly into the main code.

To find the translational motion, Eqs. (12) are simplified by:

$$\begin{aligned}
M_x &= M_y = M_z = 0 \\
u_1 \,\|\, u_2 &= F_{MAX} \\
u_3 \,\|\, u_4 &= F_{MAX} \\
u_5 \,\|\, u_6 &= F_{MAX}
\end{aligned} \tag{14}$$

where the force vector closest to the center of mass of the SSV-beam pair is the one equal to F_{MAX}. From these, symbolic solutions for the forces can be entered directly into the main code. Once these quantities are known, and the translational component necessary is found and translated to the SSV frame, the following equations are iteratively solved:

$$\begin{aligned}
&\sum F = (m_S + m_B) \cdot a = F - (C_{dvS} + C_{dvB}) * v \\
&\rightarrow a_i = (u_1 + u_2 - (C_{dvS} + C_{dvB}) * v_{i-1}) / (m_S + m_B) \\
&v_i = v_{i-1} + a_{i-1} \cdot \Delta t \\
&d_i = d_{i-1} + v_{i-1} \cdot \Delta t + 0.5 \cdot a_{i-1} \cdot \Delta t^2
\end{aligned} \tag{15}$$

G. Weighting and Final Result

Each time and fuel consumption pair is utilized by the following cost function derived from Eq. (1). This function uses simple weighing factors W_i to determine the tradeoff between fuel and time.

$$J = W_1 \cdot t_{total} + W_2 \cdot u_{total} \tag{16}$$

where W_1 is the time weighting factor, and W_2 is the fuel/power weighting factor.

Whenever a new value of J is lower than the previous stored value, the new data overwrites the old. This continues until all solved-for time-fuel consumption pairs have been compared. The lowest J value, the time and fuel consumptions, the final orientation of the vehicle, and numbers detailing which path was executed in the path planner to give this result are then output to the task scheduler.

IV. Case Study

The above trajectory planning algorithm has been implemented in C++. The specific numerical values used in the implemented code are listed in Table 1, Table 2, and Table 3.

Table 1: Physical Characteristics of Objects

	Mass (kg)	Inertia (kg*m^2)	Radius (m)	Length (m)	$C_{d\omega}$ (N*m/(rad/s)^2)	C_{dv} (kg/s)
SSV	76.2	2.7	0.3025	n/a	2.2	413.685
Beam	11.3636	0.0124, 0.0083	0.0606	3.048	changes	changes
Hub	n/a	n/a	0.1905	0.3810	n/a	n/a

Note: the beam is modeled as a cylinder with thickness 0.0063 m.

Table 2: Fixed Local Points on Objects, Local Coordinate Systems

	${}^{l}p_{contact,1}$ (m)	${}^{l}p_{contact,2}$ (m)	${}^{l}p_{contact,3}$ (m)
SSV	[0.3025 , 0 , 0]		
Beam	[0 , 0 , -1.524]	[0 , 0 , 1.524]	
Hub	[0.1905 , 0 , 0.33]	[-0.0953 , 0.165 , 0.33]	[-0.0953 , -0.165 , 0.33]

Table 3: Constants Found from Matlab Code for SSV Ramp Movement

	Full-On Ramp Up	Full-On Ramp Down	Drag-Only Ramp Down
Distance (m)	0.0204	0.0015	0.0047
Time, distance (s)	0.8800	0.1200	0.5700
Angle (rad)	4.3124	0.4169	5.9203
Time, angle (s)	4.0200	0.7400	121.6300

One case was implemented to test each part of the algorithm for the full-powered ramps, and the waypoints and results are listed in Tables 4 through 7 below. The current algorithm does not have any object avoidance built into the functions, so these movements in some cases would have skewered the vehicle on the beam; however, we are only interested in general movement and implementation at the moment, so this is negligible for now.

Table 4: Test Scenario for Algorithm – Inputs

	Initial translation (m)	Initial rotation	${}^{I}p_{contact}$, in inertial
SSV	[0 , 0 , 0]	[-1 , 0 , 0] [0 , -1 , 0] [0 , 0 , 1]	[-1 , 0 , 0]
Beam	[5 , 0 , 0]	[0 , 0 , 1] [0 , 1 , 0] [-1 , 0 , 0]	[3.476 , 5 , 0]
Hub	[6.524 , 5.3810 , 0]	[0 , -1 , 0] (z-axis of $p_{contact}$)	[6.524 , 5 , 0]

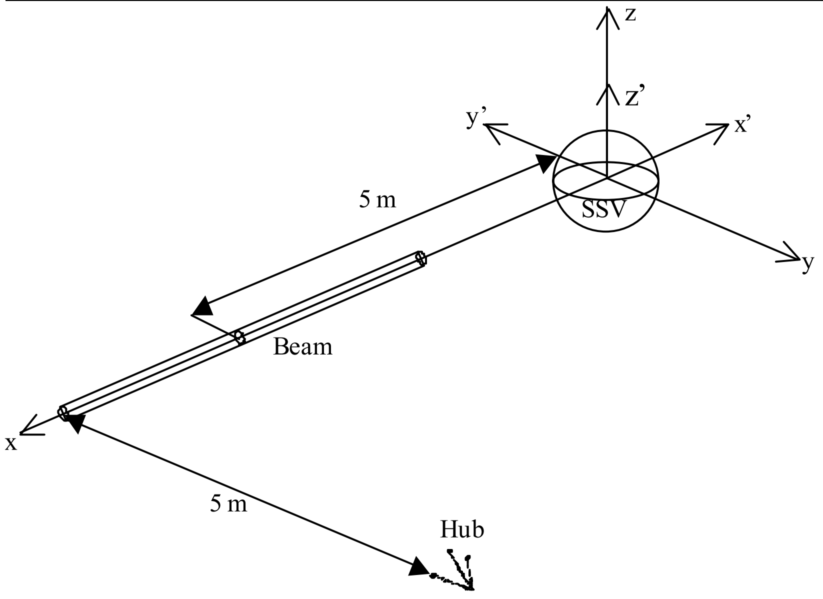

Figure 10: Test Scenario Initial Configuration

For this test case, SSV starts at the inertial space origin, rotated 180° in the x-y plane from the inertial space coordinate axis. The length of the beam is laid out along the x-axis in the inertial frame; the center of mass is 5 meters away from the origin of the inertial frame. The hub alignment axis is parallel to the y-axis in the inertial frame, and is pointing towards the end of the beam without the attach point (see Fig. 10). For each $p_{contact}$ point, the movement was solved in this way:

- ${}^{I}N_{SSV}$, the new location of the center of mass of SSV, ${}^{I}p_{CMS}$, was found. (In this case, the point was offset a distance (R_S+R_B) from the x-axis parallel to the y-axis in the inertial frame.)
- The axis of rotation and angle about which it would need to rotate, were found. (In this case, the axis of rotation was the z-axis of SSV, and the angle rotated through was 90°.)

- The translational then rotational movement of SSV was solved for iteratively.
- The thrust characteristics and time of completion were saved.

Table 5: Test Scenario for Algorithm – SSV Movement

	${}^{I}p_{contactB,1}$	${}^{I}p_{contactB,2}$	${}^{I}p_{contactB,3}$
Translation to Beam, ${}^{I}N_S$	[6.524 , -0.3025 , 0]	[5 , -0.3025 , 0]	[3.476 , -0.3025 , 0]
Rotation to Beam, *angle*, in radians	1.5708	1.5708	1.5708
Axis of rotation to Beam, ${}^{I}axis$	[0 , 0 , 1]	[0 , 0 , 1]	[0 , 0 , 1]
Time of translation, *x*-direction, in seconds	225.1518	172.6138	120.0758
Fuel of translation, *x*-direction, in Newtons	12	12	12
Time of translation, *y*-direction, in seconds	10.6733	10.6733	10.6733
Fuel of translation, *y*-direction, in Newtons	12	12	12
Time of rotation, in seconds	2.1000	2.1000	2.1000
Fuel of rotation, in Newtons	12	12	12

- The center of mass, axis of rotation, and moment of inertia of the SSV-beam pair were found. (In this case, the axis of rotation was parallel to the *z*-axis of SSV, the *y*-axis of the beam, and the *z*-axis in the inertial frame.)
- The angle that the SSV-beam pair needs to rotate through was found. (In this case, the angle rotated through was 90°.)
- The new location of the beam attach point, ${}^{I}p_{contactB}$, after such a rotation was found, and the translational distance through which it would have to move was found. (In this case, translation occurs in the *x*-*y* SSV plane only.)
- The rotational then translational movement of the SSV-beam pair was solved for iteratively.
- The thrust characteristics and time of completion were saved.

Table 6: Test Scenario for Algorithm – SSV-Beam Pair Movement

	${}^{I}p_{contactB,1}$	${}^{I}p_{contactB,2}$	${}^{I}p_{contactB,3}$
${}^{I}p_{CMpair}$	[6.3262 , -0.2632 , 0]	[5 , -0.2632 , 0]	[3.6738 , -0.2632 , 0]
Rotation to Hub, *angle*, in radians	1.5708	1.5708	1.5708
Axis of rotation to Hub, ${}^{I}axis$	[0 , 0 , 1]	[0 , 0 , 1]	[0 , 0 , 1]
Translation to Hub, ${}^{I}N_{pair}$	[-0.0654 , 2.4130 , 0]	[1.2608 , 3.7400 , 0]	[2.5870 , 5.0654 , 0]
Time of rotation, in seconds	6.7600	2.6100	6.7600
Fuel of rotation, in Newtons	12	12	12
Time of translation, *x*-direction, in seconds	0.0300	36.6300	82.3500
Fuel of translation, *x*-direction, in Newtons	12	12	12
Time of translation, *y*-direction, in seconds	85.9600	137.6500	189.2800
Fuel of translation, *y*-direction, in Newtons	10.62	10.62	10.62

- The final numbers were combined and tabulated; see Table 7.

Table 7: Test Scenario for Algorithm – Total Cost

	${}^{I}p_{contactB,1}$	${}^{I}p_{contactB,2}$	${}^{I}p_{contactB,3}$
Time of translation, SSV, in seconds	225.1518	172.6138	120.0758
Fuel of translation, SSV, in Newton-seconds	2.8299e+003	2.1995e+003	1.5690e+003
Time of rotation, SSV, in seconds	2.1000	2.1000	2.1000
Fuel of rotation, SSV, in Newton-seconds	25.2000	25.2000	25.2000
Time of rotation, pair, in seconds	6.7600	2.6100	6.7600
Fuel of rotation, pair, in Newton-seconds	81.1200	31.3200	81.1200
Time of translation, pair, in seconds	85.9600	137.6500	189.2800
Fuel of translation, pair, in Newton-seconds	913.2552	1.9014e+003	2.9984e+003
Total time, in seconds	319.9718	314.9738	318.2158
Total fuel, in Newton-seconds	3.8495e+003	4.1574e+003	4.6737e+003

From the data, it looks as though the open point on the beam, farthest from the origin of the inertial frame, is the optimum point to attach to and move from if fuel use drives the cost function; if time drives the cost function, the center of the beam looks to be the best choice – it was somewhat of a intuitive toss-up between that point and the point at the beam's center of mass, since rotating about the center of mass caused less drag, but the extra distance and suboptimal drag characteristics of its translational movement more than used up those savings.

V. Conclusion

This paper describes an architecture for determining an assembly procedure for autonomous space or underwater structural assembly, focusing on the problem of solving individual assembly task trajectories and their associated costs. These assembly task trajectories are split into two main subdivisions of movement: vehicle to object, and vehicle-object pair to destination. Each of these movements is decoupled so that the translational and rotational motion occur separately, to avoid complex dynamics. The specific algorithms for each movement are presented; these are solved iteratively over several waypoints and differing magnitudes of thrust. A cost function is also presented that uses the thrust characteristics and time to completion to compute a total cost J that is useful to a task scheduler. This particular trajectory planner and cost function is an implementation to solve a very constrained case. However, the simpler cases have been tested and do seem to be reasonable, as the case study section shows.

Future additions and refinements to this planner after the basics have finished being implemented and tested could include other types of combined SSV-beam movement (translation then rotation; rotation to minimum drag configuration, translation, then final rotation to correct orientation), object avoidance, velocity caps, attention to the walls of the tank, and coupled motion. Better weighting factors, and their refinement to determine which combinations give more desirable outcomes, is also another improvement that should occur down the road prior to implementation, or possibly in conjunction with neutral buoyancy testing. More complex dynamics problems associated with this type of construction scenario – movement of partially-constructed assemblies to other locations, navigation of a SSV-beam pair between objects, vibrational and nonrigid body motion – are quickly encountered when certain assumptions are relaxed. A generalized, further extended planner that can model these sorts of movements would be a short step away from a final version of a path planner that could be utilized in an actual complex construction scenario.

Acknowledgments

I would like to thank Dr. Ella Atkins for her support as my mentor and advisor in this work. I would also like to thank Jenn Volpe for helping along my brainstorming sessions and the Space Systems Lab students for their ideas and the use of their facilities.

References

[1]Nau, D. S., Smith, S. J. J., Erol, K., "Control Strategies in HTN Planning: Theory versus Practice," *Proceedings of the Fifteenth National Conference on Artificial Intelligence, Tenth Conference on Innovative Applications of Artificial Intelligence*, AAAI Press, Menlo Park, CA, 1998, pp. 1127-1133.

[2]Viggh, H., "Artificial Intelligence Applications in Teleoperated Robotic Assembly of the EASE Space Structure," M.S. Thesis, Aeronautics and Astronautics Dept. and Electrical Engineering and Computer Science Dept., Massachusetts Institute of Technology, Cambridge, MA, 1988.

[3]Hossaini, L. S., "The Design and Analysis of a Second Generation Free Flying Underwater Camera Platform," M.S. Thesis, Dept. of Aerospace Engineering, University of Maryland, College Park, MD, 2000.

[4]Birkhoff, G., *Hydrodynamics, A Study in Logic, Fact, and Similitude*, Princeton University Press, Princeton, NJ, 1960, pp. 16-17.

[5]Dally, J., and Bonenberger, R., Jr., *Design Analysis of Structural Elements*, 2nd ed., College House Enterprises, LLC, Knoxville, TN, 2000, pp. 376.

[6]Hibbeler, R. C., *Engineering Mechanics Dynamics*, 9th ed., Prentice-Hall, Upper Saddle River, NJ, 2001, pp. 546-549, 562.